OpenStack Keystone Federation
Enabling Interoperability Across Clouds

Marek Denis
Research Fellow,
CERN
OpenStack ATC

Steve Martinelli
IBM Software Developer, OpenStack ATC
Keystone Core

Joe Savak
Senior Product Manager, Rackspace
OpenStack ATC

Brad Topol
IBM Distinguished Engineer, OpenStack
OpenStack ATC

A lot of individuals and companies have contributed to making Federation a core feature provided by Keystone.

Many thanks to:

redhat

CERN

University of Kent

IBM

What is OpenStack & Keystone?

OpenStack:

An Open Source Cloud Operating System which allows implementors to:

-- Provision and manage compute, network, and storage resources quickly

-- Monitor and alert on those resources

-- Auto-scale cloud resources to satisfy traffic needs

-- Standardize and control disk & server images

Keystone:

The Identity service that comes bundled with OpenStack. Keystone allows implementors to:

-- Provision identities and manage their authorization

-- Integrate with an existing directory via LDAP or use a variety of other back-end options

-- Programmatically discover implemented cloud services

A bit of history:

OpenStack was jointly launched by Rackspace and NASA in July 2010 to offer cloud computing services. Since then it has been adopted by over 20 cloud service providers, and has 124 company contributors for the latest release.

Federated Cloud Definition

*Pooling of resources from disparate, potentially heterogenous, cloud systems where interoperability and portability enable sharing, migration, and redundancy which is all ensured through a common mechanism (such as central management system or a common API), **and where identity and authorization management mechanisms are established**.*

With Keystone, a single identity can perform operations across multiple clouds through established authorization mechanisms backed by standard federation protocols.

	OpenStack Keystone
Amazon Web Services IAM	✓
Microsoft Azure	✓
VMWare	✓
Google Cloud	✓

What makes federation in Keystone special?

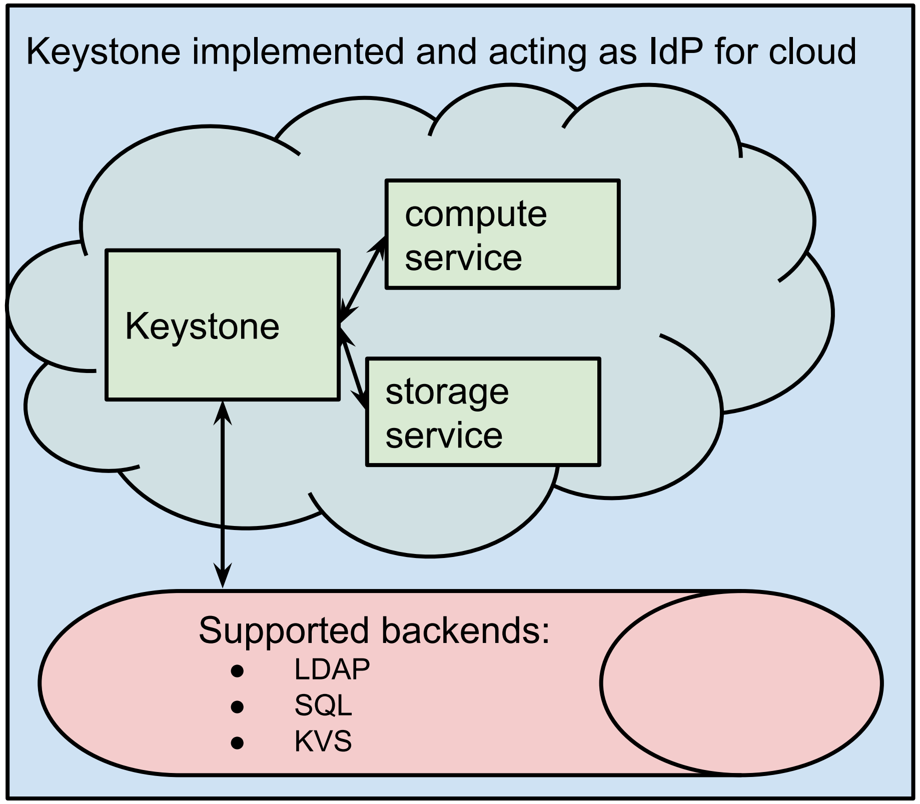
OpenStack services follow Service Oriented Architecture. This includes:

- Independence from any vendor or product
- A focus on interoperability over custom integrations

For the federation implementation, this results in:

- No “favorite” federation protocol
- No “favorite” identity backend
- No “favorite” service provider
- Extensible attribute mapping functionality and APIs
- Discoverability - configured identity providers & services providers, mappings, and federation protocols can be programmatically discovered

Typical Keystone implementation (before federation)



Federation Use Cases

Multiple Cloud Service Providers

Implementor would like to allocate workloads between multiple cloud service providers without having to maintain identities in each service provider.

Cloud Bursting

Implementor would like to auto-scale infrastructure to service-providers without having to maintain identities in each service provider.

Easy federation workflows for clients

Client developers would like to be aware of as few federation protocols as possible.

Easy to configure

Customers of cloud service providers can easily integrate their existing identity provider with their cloud accounts.

Non-Keystone service providers

Implementor would like to use the keystone service-provider federation solution to connect to service-providers that don't run keystone, but do run a supported federation protocol.

Central policy information point for service provider trusts

Implementor would like on-premises keystone to be the single system for handling service provider trusts.

Identities federating in

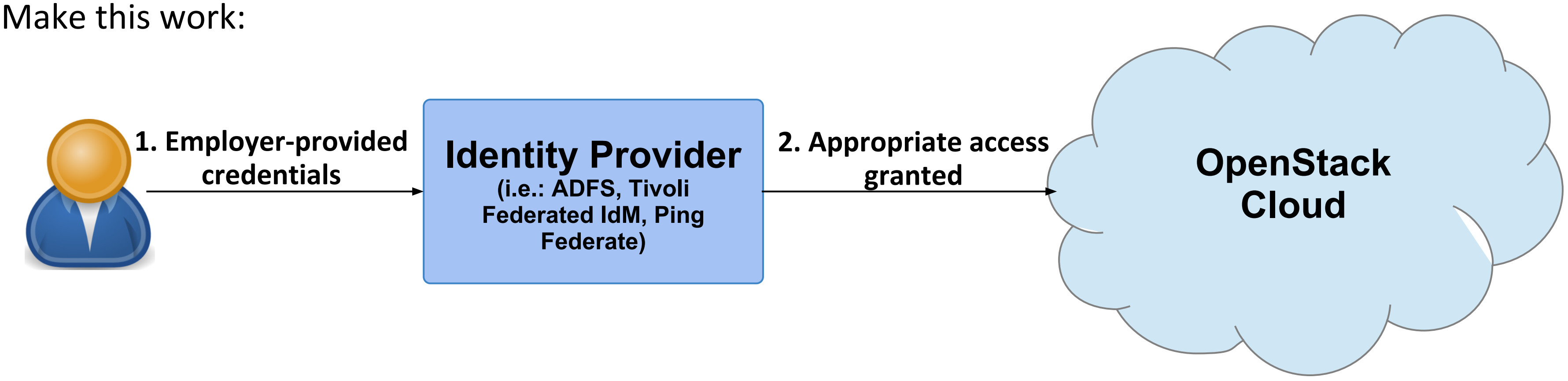
Allow easy integration with existing identity providers through the use of federation protocols.

Identities federating out

Allow easy integration with services providers through the use of federation protocols

Icehouse Release: Federate In!

Make this work:


1. Employer-provided credentials
Identity Provider
(i.e.: ADFS, Tivoli Federated IdM, Ping Federate)
2. Appropriate access granted
OpenStack Cloud

As an OpenStack implementor, allow me to use an identity provider I have in place today that supports federation protocols to integrate with OpenStack clouds.

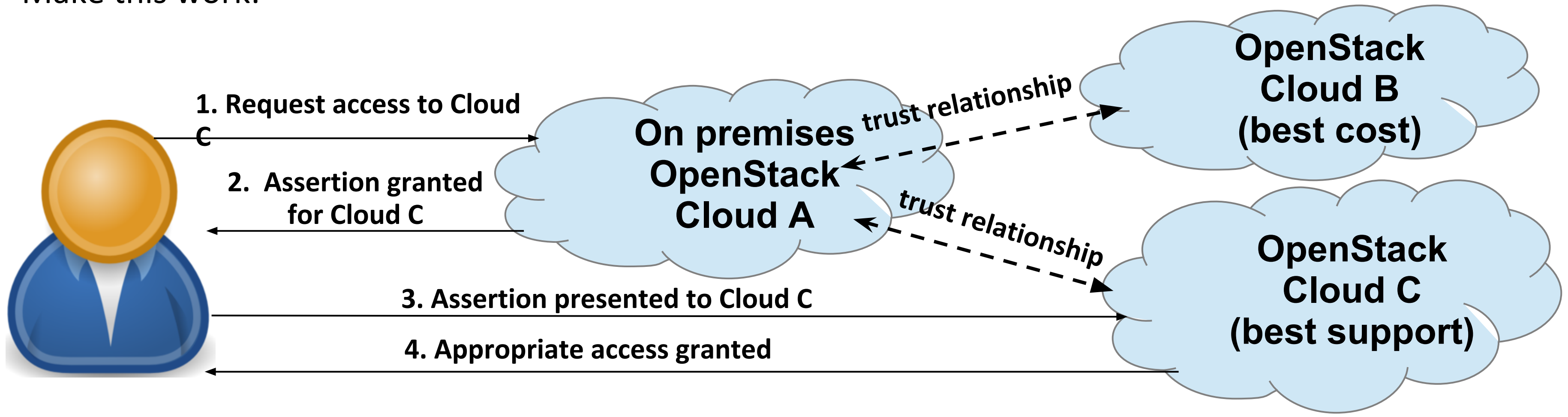
Benefits:

- Control access centrally to multiple service providers (including multiple OpenStack Clouds).
- Allow users to use one credential to have access to multiple services (single sign-on).
- Removes identity provisioning and maintenance steps required for pre-icehouse releases of OpenStack.

Currently supporting SAML. OpenID connect in the works.

Juno Release: Federate Out!

Make this work:


1. Request access to Cloud C
2. Assertion granted for Cloud C
3. Assertion presented to Cloud C
4. Appropriate access granted
On premises OpenStack Cloud A
trust relationship
trust relationship
OpenStack Cloud B (best cost)
OpenStack Cloud C (best support)

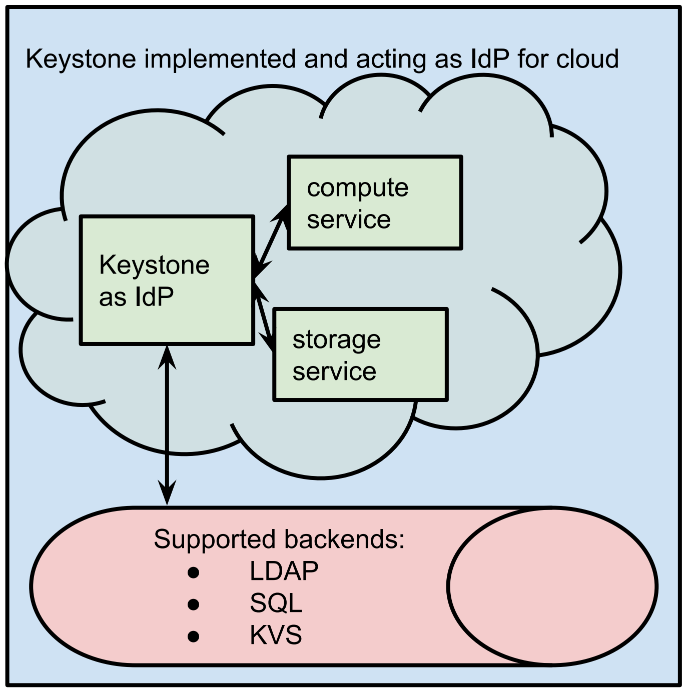
As an OpenStack implementor, allow me to use my OpenStack Cloud I have in place to manage trusts with other OpenStack clouds or non-OpenStack service providers.

Benefits:

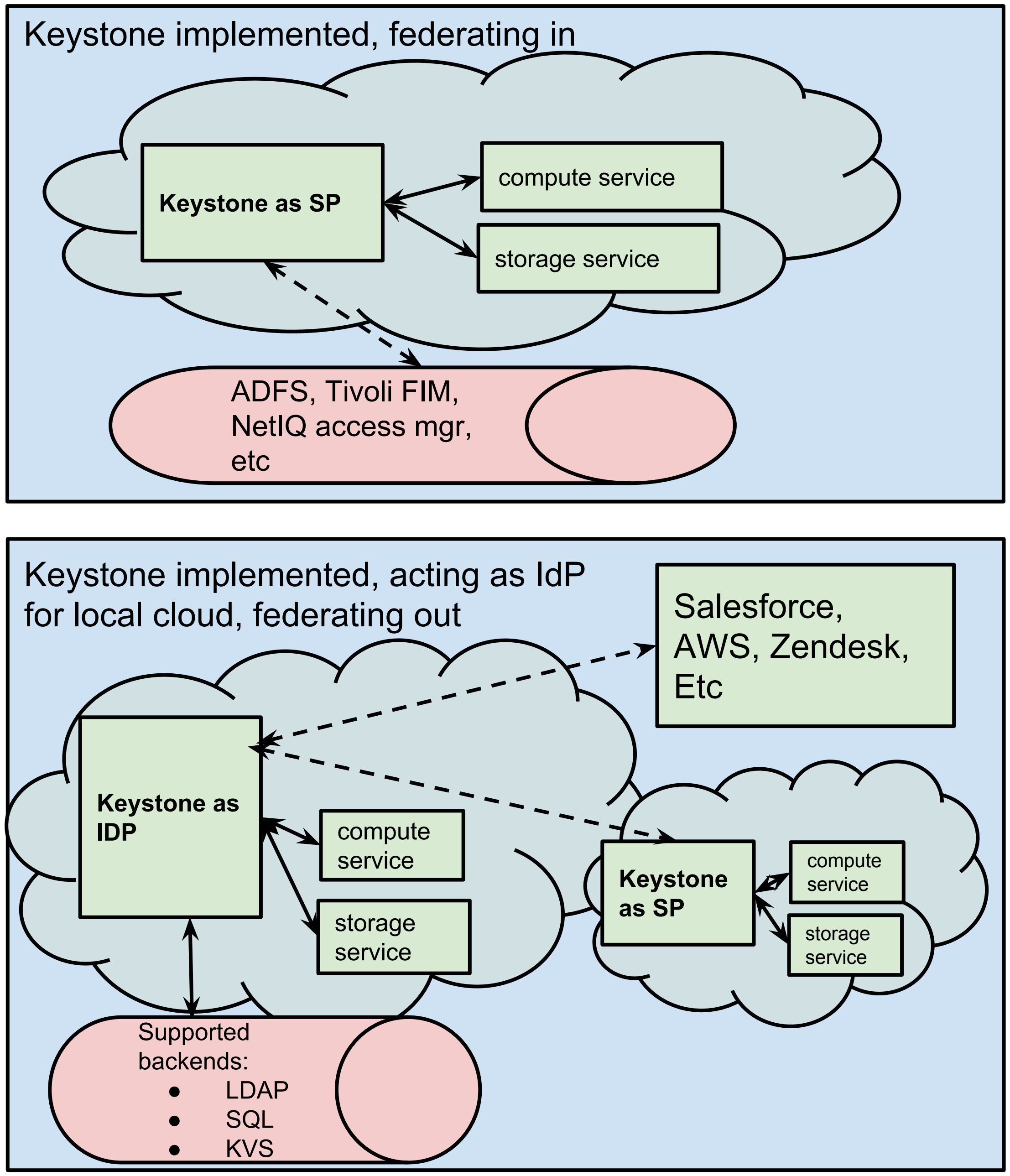
- Clients coded against one OpenStack cloud need to know only the authentication or federation protocol for that one cloud to access another.
- Management of authorization data centrally keeps governance
- Cuts out the need to provision identities in each integrated OpenStack cloud or service-provider.

Currently supporting SAML.

Traditional implementation



Federation implementation options


Keystone implemented, federating in
Keystone as SP
compute service
storage service
ADFS, Tivoli FIM, NetIQ access mgr, etc
Keystone implemented, acting as IdP for local cloud, federating out
Salesforce, AWS, Zendesk, Etc
Keystone as IDP
compute service
storage service
Keystone as SP
compute service
storage service
Supported backends:
● LDAP
● SQL
● KVS

How does authorization work?

Authorization data is split between the identity provider and the service provider:

Identity Provider is responsible for :

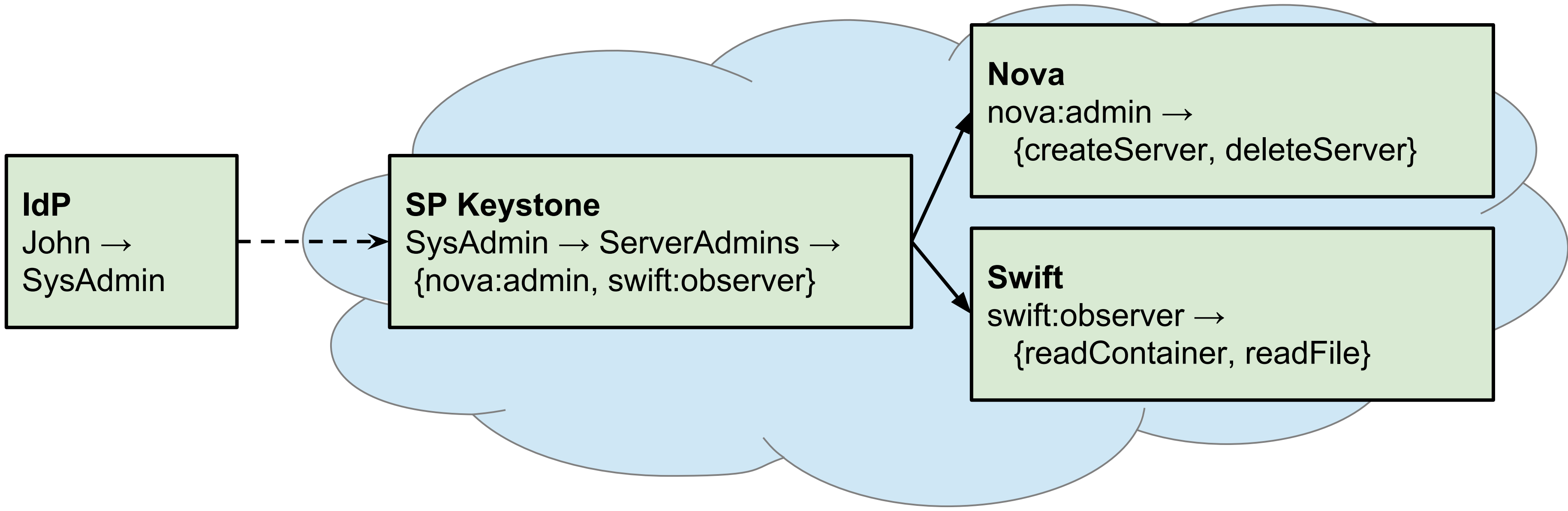
- User → Group(s)
- Groups are sent over as part of the signed assertion to the service provider.

Service Provider is responsible for:

- After verifying the assertion, applying the mapping:
  - Group(s) that the IdP sent → Group(s) that the SP understands → Role(s)

The service provider's endpoint service is responsible for:

- Role → Capability


IdP
John →
SysAdmin
SP Keystone
SysAdmin → ServerAdmins →
{nova:admin, swift:observer}
Nova
nova:admin →
{createServer, deleteServer}
Swift
swift:observer →
{readContainer, readFile}

Kilo Release and beyond, a look forward

Attribute mapping enhancements

- Rather than listing group IDs, maybe something more dynamic, could make the deployers life easier.

Fine grained access control

- Restrict access to specific resources (servers, networks, etc). The concept of “resource groups” can help facilitate the grouping of resources (ex: “prod servers”, “dev environment”) to aid in access control management.

Support additional federation protocols

- For the Service Providers, just install new apache plugins (like *mod_openid, mod_auth_kerb*).
- For Identity Providers, a bit more complex, would need to, for example, generate the OpenID Connect token.
- Other folks already have working prototypes with *mod_mellon, mod_abfab* and *mod_auth_kerb.*

Questions?